DECISION MAKING OF A MOBILE ROBOT IN THE PRESENCE OF RISK IN ITS SURROUNDINGS

A Thesis

by

SUNG HUH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2011

Major Subject: Mechanical Engineering

DECISION MAKING OF A MOBILE ROBOT IN THE PRESENCE OF RISK IN ITS SURROUNDINGS

A Thesis

by

SUNG HUH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Reza Langari
Committee Members,	Sivakumar Rathinam
	Dylan Shell
Head of Department,	Jerald A. Caton

December 2011

Major Subject: Mechanical Engineering

ABSTRACT

Decision Making of a Mobile Robot in the Presence of Risk in Its Surroundings.

(December 2011)

Sung Huh, BS, Texas A&M University

Chair of Advisory Committee: Dr. Reza Langari

Mobile robots are used on many areas and its demand on extreme terrain, hazardous area, or life-threatening place is increasing to reduce the loss of life. A good decision making capability is essential for successful navigation of autonomous robot and it affect finding the shortest or optimal path within given condition. The wavefront algorithm is simple to apply, yet yield an optimal path for a robot to follow in many different configurations. Although the path created using wavefront algorithm is an optimal in the sense that every node has the same cost, the result is not the best result in global perspective because of the algorithm is inconsiderate on the surrounding condition.

To solve this issue and create the best result on global perspective, risk factor analysis method was implemented on the wavefront algorithm to improve the performance. In this work, the relationship between the wavefront algorithm and dynamic programming will be explained to show that the wavefront algorithm obeys the principle of optimality. The simulation result displays better performance on safety, while keeping the traveling distance minimum, if the risk factor is used on the

wavefront algorithm and the robot on actual test behave accordingly. While keeping the distance minimum, method using risk factor produces path that keeps certain distance around the obstacle, thereby enhances safety while preserving optimality of path. This work will contribute on decision making of mobile robot using risk factor method to create a most desirable and safe path. In addition to that, it will demonstrate how the risk factor method can be applied to the mobile robot navigation.

DEDICATION

To my mother, father, and sister

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Langari, for his guidance throughout the course of this research, and my committee member Dr. Rathinam and Dr. Shell on their generous and thoughtful feedbacks on my works.

I would like to give my gratitude to my friends, colleagues, and everyone else who helped achieving my academic success, and sharing a joyous time throughout my academic career.

Finally Special thanks go to my mother, father, and sister for their relentless encouragement, support, and patience on my education.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Mobile robots are used on many areas and continue to take important roles on our lives. The purposes of mobile robot include, but not limited to exploring hazardous and dangerous site, search and rescue, reconnaissance, etc. One of the most well known applications is the Mars Rover, where a mobile robot explores Mars and collects data. In addition to the Mars Explorer, the military uses mobile robots on scouting enemy's whereabouts, removing landmine from mine fields [1], and many more. Its primary objective is to work in an area that is dangerous for humans to work and to reduce the risk of loss of human life.

A well developed navigation skill is important for an autonomous robot to have an independent decision making capability. In order to improve the navigation function of an autonomous mobile robot, researchers have developed many strategies that utilize terrain conditions and sensors such as vision, GPS, laser range finders for path planning, and incorporate available information to develop a path for robot. The general motion planning problem involves finding a collision-free motion of the robot from the initial state to the final state with satisfying any constraints [2].

One of famous methods of path planning is the potential field methodology developed by Khatib [3]. This method treats any objects and goal as a source of potential field on the map, and uses resulting vector field of potential field vectors as a guiding path for a robot to reach the goal. While potential field method generates set of vector

---

This thesis follows the style of *IEEE Transactions on Robotics.*

fields on the map, motion planning in general uses map represented by finite number of nodes or grids that contains important feature on the field. Increasing number of grid result in better representation of the map thus increases performance of robot as well. Notable examples are car navigation system [4] and routing algorithm on online web mapping service [5], where driver receive the fastest path between the designated locations. With utilizing advanced sensor and computer system, mobile robot can navigate unknown terrain effectively without human supervision.

The wavefront algorithm is simple to apply and yield an optimal path. Although the path created using wavefront algorithm is an optimal since the every node has the same cost of travelling, the result is not the best result in global perspective because of the algorithm is inconsiderate on the surrounding condition. The potential field method is one of method that creates a path with presence of obstacle, and applying the concept from potential field into the wavefront algorithm produces a path that is optimal and avoids obstacle collision as much as possible. However, the wavefront algorithm and the potential field method have different way of representing map and path creation. To resolve this difference and assimilate two different methods to create the best result on global perspective, risk factor analysis method was implemented to improve the performance.

Contribution

The major contribution of this research is developing mobile robot path planning using risk factor method that is used on a decision making of a project management as a part of risk analysis. As a part of application and analysis, the relationship between the

Principle of Optimality, Dijkstra's Algorithm, and the wavefront algorithm will be explained, and explain how new method satisfy the Principle of Optimality and the resulting path is optimal in the Dynamic Programming perspective.

Outline

The organization of this thesis is as follows. The second chapter discusses apparatus and experimental equipment used in this research. The third chapter explains theoretical background on motion planning problem, including Bellman's Principle of Optimality, Dijkstra's algorithm, the wavefront algorithm, and potential field method.

The fourth chapter will be on risk factor analysis. Background, theory, and its application on mobile robot path planning will be discussed. The fifth chapter will show the simulation and test result of new method.

The sixth chapter discusses relationship of each existing algorithm with respect to the Dynamic Programming, demonstrate how new method satisfies the Principle of Optimality, and its result is optimal in Dynamic Programming perspective. The Conclusion will discuss the result of research and potential future work.

CHAPTER II

APPARATUS AND EXPERIMENTAL SETUP

Mobile Robot Platform

Figure 1 is the programmable mobile robot used in this research, which is known as iRobot Create® developed by iRobot® Company for researcher, hobbyist, or educator. It can be programmed using C or C++ and communicate via serial cable or Bluetooth wireless connection. Additional sensor or electronic devices can be attached through 25 pin Cargo Bay Connector. Its motion is determined by controlling speed of each motor. Its diameter, minimum, and maximum speed is 0.33 m, 0.1m/s, and 0.5 m/s, respectively. iRobot Create® is equipped with a pack of sensor for its basic function. These sensors are wheel encoder, wall sensor, omnidirectional IR sensor, cliff sensor, and bumper sensor. However, none of these sensors are capable of measuring distance between robot and obstacle. In order to include distance measuring capability on iRobot Create®, attaching an external sensor is mandatory.

Figure 1. Programmable mobile robot plat form (iRobot Create®) used in the experiment.

Sensor

Figure 2 is a selected sensor for distance measurement, Sharp® GP2D12 infrared sensor, which is cost effective and small size sensor for measuring the distance compare to other expensive distance measuring sensors. It is a sensor that uses triangulation and a small linear CCD array to compute the distance and presence of objects in the field of view [6]. It is operated by emitting a pulse of IR light in the field of view and reflects back to a receiver if light hits object. The linear CCD array enclosed in the receiver calculates angle the reflected light came back, and calculate the distance to the object using this angle [6]. Its range of operation is 10 – 80 cm, as can be seen in Figure 3. Sharp® IR sensor is available on both analog and digital type.

The output of the sensor is an analog voltage that has nonlinear characteristic with respect to the distance, and this output is converted using 10 – bit analog to digital port on the Cargo Bay Connector.

Figure 2. GP2D12 analog distance measuring sensor.

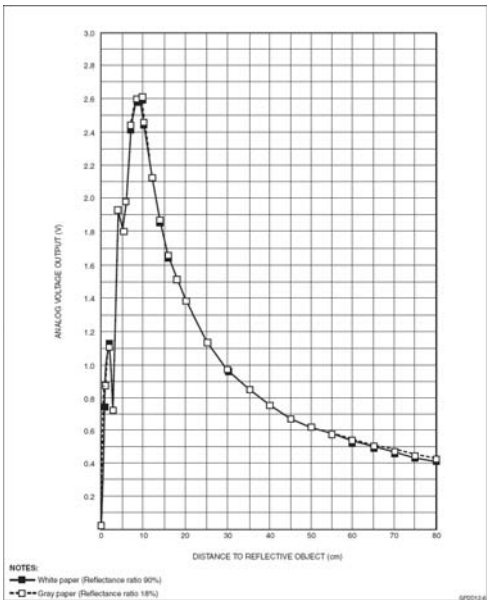


Figure 3. GP2D12 measurement range from datasheet [7].

Communication

Figure 4 is a Bluetooth module used for communication between robot and hosting computer. Program code is transferred from host computer to the robot by direct tethering serial communication port to mini-DIN connector or wirelessly using Bluetooth Adaptor Module (BAM®) from Element Direct®. Tethering robot has

guaranteed connection between robot and computer, but has disadvantage of physically connecting robot and computer.

On contrary, wireless connection via Bluetooth eliminates this wire tethering while establishing communication between computer and robot. Bluetooth dongle used in this research is Class 1 device, which has wireless range of 100 m (330 ft) from the host computer. Therefore, wireless communication method is preferred over tethering communication and suitable for mobile robot experiment.

Figure 4. Bluetooth Adaptor Module for iRobot Create®.

Programming Environment

Programming iRobot Create® is done by MatLAB®. Esposito and Barton developed a suit of function that allows user to program iRobot Create® directly in MatLAB® environment [8]. In addition to their works, additional function that uses I/O port on iRobot Create® is developed for external sensor. The specification of the computer is Intel Pentium® 4 1.5GHz processor, 512 MB RAM with Microsoft® Windows XP SP3 based operating system, and the version of MatLAB is 7.0.4.

CHAPTER III

MOTION PLANNING AND THE SHORTEST PATH PROBLEM

One of the most important functions of a mobile robot is a capacity of navigating the environment autonomously. A success of navigation depends on robot's ability to represent surrounding environment, locate its position with respect to any obstacles, and create a path from its position to the goal safely [9]. The first two criteria are widely used by SLAM (Simultaneous Localization And Mapping) to achieve its objective and the last criterion is achieved by a path planning. This path planning involves generating path with given constraints and it has been researched extensively. The challenge is to create a best path to the destination, where best path can be a collision free or the minimal distance. The shortest path problem is finding the path that has the shortest distance from current point to the destination point among all possible intermediate point. There are numerous algorithms that find the shortest path and most of them display similar property. These algorithms have property of what is known as the Principle of Optimality, which is the fundamental of the Dynamic Programming.

Bellman's Principle of Optimality

The theory on the Principle of Optimality was developed by Richard Bellman in 1954. It is the principal theory of Dynamic Programming that used on optimizing various multistage decision processes. Bellman stated in his publication that "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting

from the first decision [10].” In other words, According to LaValle’s interpretation [11], a portion of an optimal plan must be an optimal itself and there must not any other optimal plan, and any portion of an optimal trajectory from an intermediate state to the final state is itself the optimal trajectory from the intermediate state [12].

The Dynamic Programming is widely used on optimizing deterministic process. One of the most well known applications of Dynamic Programming is the shortest path problem [13]. Figure 5 is schematics of node-link network used for the shortest path problem and Figure 6 is possible node configurations in the network suggested by Powell [13].

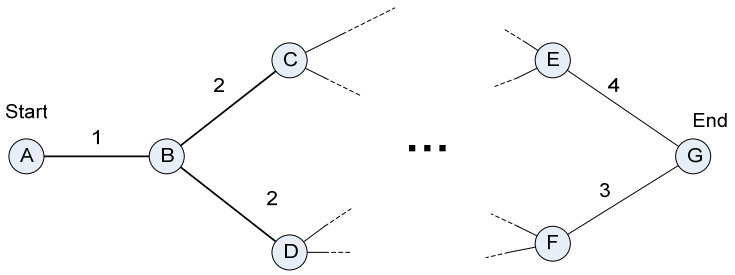


Figure 5. Network of node and links for the shortest path problem.

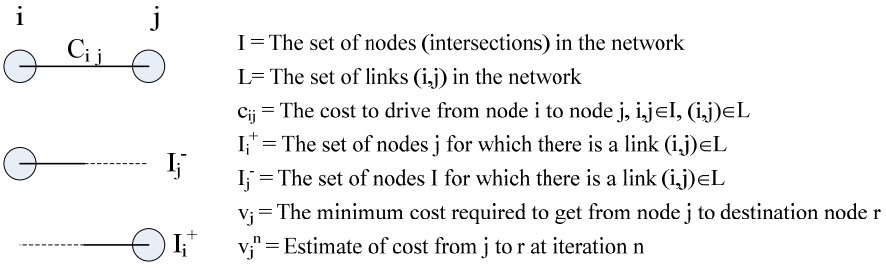


Figure 6. Configuration of nodes in the network.

The following procedure is the shortest path using Principle of Optimality explained by Powell [13]

$$\text{Let } v_j^0 = \begin{cases} M & j \neq r \\ 0 & j = r \end{cases}, \text{ M is some big number and n =1.}$$

1. Solve for all i $\in$ I

$$v_i^n = \min_{j \in I_i^+} \left(c_{ij} + v_j^{n-1} \right) \tag{1}$$

2. If $v_i^n < v_i^{n-1}$ for any i, let n = n+1 and repeat the process. Otherwise, stop the iteration.

Equation (2) is the shortest path problem form of Principle of Optimality, often referrend to as Bellman's algorithm [13], and it is governing equation for many different shortest path algorithms. One of variation of Bellman's algorithm is famous Dijkstra's algorithm, which chooses the node from the candidate list, or priority queue, with the smallest value of the cost.

Motion Planning Problem

The shortest path problem, when applied to mobile robot navigation, can be treated as a planning problem. This planning problem, according to Russell and Norvig [14], is "the task of coming up with a sequence of actions that will achieve a goal." The general motion planning problem of mobile robot is described as finding a collision-free motion of the vehicle from the initial state to the final state with satisfying any given constraints [2]. LaValle [11] stated that the discrete feasible plan of motion planning problem consists of following components:

- A nonempty state space X
- A finite action space $U(x)$ for each state $x \in X$

- A state transition function f that produces a state, $f(x,u) \in X$, for where $x \in X$ and $u \in U(x)$.
- An initial state $x_I \in$ X
- A goal set $X_G \subseteq X$

If a solution from discrete feasible plan is to optimize some criterion in addition to finding the sequence of actions that leads to the goal set, it becomes the discrete optimal planning and three more additional components are added to the discrete feasible plan: A stage index to indicate current plan step, cost function, and termination action to stop the plan and fix the total cost. More specifically, according to LaValle [11] optimal planning problem contains following properties in addition to the feasible planning problem,

- A number of stages, which is the length of a plan measured as the number of actions $u_1, u_2, \ldots, u_K$.
- A stage–additive cost function L, which is applied to a K–step plan, π_K.

$$L(\pi_K) = \sum_{k=1}^{K} l(x_k, u_k) + l_F(x_F)$$

- Each $U(x)$ contains the special termination action, u_T.

where F is defined as the final statge F = K+1, and the cost term $l(x_k, u_k)$ yields a real value for every $x_k \in X$ and $u_k \in U(x_k)$, and the final term $l_F(x_F)$ is outside of the sum and is defined as $l_F(x_F) = 0$ if $x_F \in X_G$, and $l_F(x_F) = \infty$ otherwise. The optimal path is

retrieved by minimizing the cost function and it is achieved by the Principle of Optimality, which can be performed by using Dijkstra's Algorithm [11].

This plan π can be used as a navigation function via expressing as a potential function. For a potential function that maps every state from 0 to ∞ (ϕ: $X \rightarrow [0,\infty]$), the plan can be used as a navigation function by minimizing resulting potential of a state obtained using potential function ϕ [11].

LaValle [11] stated that the most desirable potential function is one that causes arrival in X_G for any initial state, and it is called a navigation function if the following requirements are satisfied

- $\phi(x) = 0$ for all $x \in X_G$
- $\phi(x) = \infty$ if and only if no point in X_G is reachable from x
- For every reachable state $x \in X \setminus X_G$, the local operator produces a state x' for which $\phi(x') < \phi(x)$.

In addition to the above requirements, the navigation function must satisfy the principle of optimality to become an *optimal navigation function* [11].

$$\phi(x) = \min_{u \in U(x)} \{ l(x,u) + \phi(f(x,u)) \} \quad (2)$$

Dijkstra's Algorithm

Dijkstra's algorithm is a special form of dynamic programming that can be used to find an optimal plan in planning problem [11], which was published in 1959 by Edsger Dijkstra in *Numerische Mathematik* [15]. In his publication, Dijkstra suggested problems on minimum length spanning tree and the minimum length between two given

nodes, and explained the method of solving each problems [15]. The solution to the second problem is known as Dijkstra's algorithm on the single-source shortest path problem for a graph and used to find the shortest path. While having a property of the Dynamic Programming explained earlier, Dijkstra's algorithm has additional characteristics that distinguishes it as an application of the Dynamic Programming. According to LaValle [11], Dijkstra's algorithm has following properties

- Non-negative cost on the edge of graph
- Priority queue is sorted according to the cost function
- The optimal cost from the initial state obtained by summing cost of travel from all possible path
- Path with the lowest cost is the optimal cost

The procedure for the algorithm is similar to Bellman's algorithm in the Dynamic Programming explained in the previous section. Following pseudo code is how Dijkstra's algorithm is used to find the shortest path in the network [16].

1. Set the value of initial node zero and make it the current node. Set values of infinity to all other nodes and mark them *unvisited*
2. If sum of the value of the current node and the value of the edge is less than the value of the adjacent node, change the value of the adjacent node to this value for each unvisited node adjacent to the current node. Leave the value unchanged otherwise.

3. Mark the current node as *visited*, and if there are unvisited nodes, Set the unvisited node with the smallest value as the new current node and repeat the process until every node is visited

Figure 7 is the generalization of Dijkstra's algorithm explained by LaValle [11]. In this procedure, x is a state, u is a corresponding action, $f(x,u)$ is a state transition function, $l(x,u)$ is a cost to apply action u from state x, and Q is a queue. Line VII and VIII correspond to the second step of pseudo code explained earlier, and C(x') in both steps represents the lowest cost to travel known so far. This relation is an indication of Dijkstra's algorithm following the Principle of Optimality, thus optimal in the Dynamic Programming perspective. The example on the usage of Dijkstra's algorithm can be found on Appendix A.

```
I.    Set C(x) =  for all x ∈ x_I, and set C(x_I)=0
II.   Q.Insert(x_I)
III.  While Q not empty do
IV.       x ← Q.GetFirst()
V.        Forall u ∈ U(x)
VI.           x' ← f(x,u)
VII.          If C(x)+l(x,u) < min{C(x'), C(x_G)} then
VIII.             C(x') ← C(x)+l(x,u)
IX.               If x' ≠ x_G then
X.                    Q.Insert(x')
```

Figure 7. Generalization of Dijkstra's algorithm by LaValle [11].

Wavefront Algorithm

Wavefront propagation algorithm is used on mobile robot path planning for its simplicity. LaValle stated that the wavefront algorithm "can be considered as a special

case of Dijkstra's algorithm that avoids explicit construction of the priority queue [11]" and Varshavskaya [17] stated that it involves a breadth-first search that examines all the nodes backwards beginning at the goal until it reaches the start point. According to Muhammad [18], Breadth first search algorithm is "a way to find all the vertices reachable from a given source vertex." In this algorithm the goal set is assigned a value of 0 and value of i is assigned into a free node at i^{th} iteration. If the nearest node is not assigned any value or not an obstacle, then assign a value of i+1 and proceed until every node is filled. More formally, the pseudo code of the wavefront algorithm by LaValle [11] is follow

1. Initialize $W_0 = X_G; i = 0$
2. Initialize $W_{i+1} = \varnothing$
3. Fore every $x \in W_i$, assign $\phi(x) = i$ and insert all unexplored neighbors of x into W_{i+1}
4. If $W_{i+1} = \varnothing$, then terminate; otherwise, let *i*:=*i+1* and go to step 2

where W_0 is the initial wavefront state W_i is the wavefront at each state *i*. The process continues until every reachable state is reached. LaValle [11] stated that the wavefront algorithm can be viewed as a special case of Dijkstra's algorithm that "avoids explicit construction of priority queue." Since the cost of travel is the same every direction in the wavefront algorithm, the cost of travel produced by cost function is the same every iteration. Therefore, at any location on the field path is an optimal in the Dynamic Programming perspective. The demonstration, example, and programming code of the

wavefront algorithm are well explained at Society of Robots [19]. Following example is a demonstration of the wavefront algorithm on path planning.

1. Locate the start and goal points on the map. Goal has the lowest potential, thus its value is zero. See Figure 8. Graphical representation of map using grid for the wavefront algorithmFigure 8.

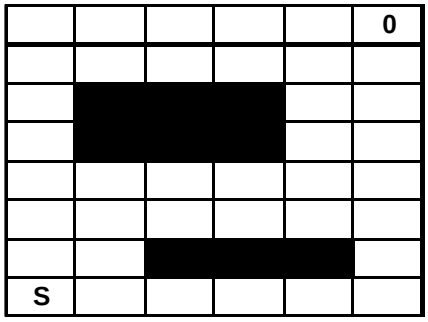


Figure 8. Graphical representation of map using grid for the wavefront algorithm.

2. To create the wavefront, set the node a value of every empty neighboring node of the current node to the value of that node+1. Repeat this process until wavefront reaches robot start location. See Figure 9.

5	4	3	2	1	**0**
	5	4	3	2	1
				3	2
				4	3
				5	4
					5
S					

Figure 9. Assignment of number with wave expansion.

3. Follow the shortest path by taking the smallest node value from the starting point. See Figure 10.

5	4	3	2	1	**0**
6	5	4	3	2	1
7				3	2
8				4	3
9	8	7	6	5	4
10	9	8	7	6	5
11	10				6
S	11	10	9	8	7

Figure 10. The shortest path (green cell) created by wavefront algorithm.

Variations of the wavefront algorithm

The Gradient Method

Other researchers developed an algorithm based on wavefront algorithm. Konolige [20] developed an algorithm that calculates an optimal path to a gal called the gradient method. Konolige stated that the cost of path is a function of path F(P) and assume that the cost is determined by adding the *intrinsic cost* of being at a point and the *adjacency cost* of moving from one point to the next.

$$F(P) = \sum_i I(p_i) + \sum_i A(p_i, p_{i+1}) \tag{3}$$

where $I(P_i)$ and $A(P_i, P_{i+1})$ are intrinsic cost and adjacent cost, respectively. The optimal path is determined by minimizing the cost of path function at k, $F(P_k)$, or mathematically,

$$N_k = \min_{P_k} F(P_k) \tag{4}$$

At the beginning of the algorithm, the goal is assigned with zero and put in an active list, while every other point is assigned with infinite cost. This active list is updated by replacing high cost with low cost, and by taking a cell with minimum cost path, robot can reach the goal from any arbitrary point [20].

In the experiment, the gradient method is successful at efficient high-speed control of a mobile robot in an uncertain environment [20]. When compare to the human operator on both known and unknown environment, the gradient method shows a significantly better performance.

Potential Field

Usage of potential function for path planning was first published by Khatib [3]. A potential field method uses the idea of assigning a potential function to the free space, and treating the vehicle as a particle reacting to forces due to the potential field. In this method, a goal has the lowest potential and always attracts the robot, while obstacles emanate repulsive forces and repel the robot away. The potential field method serves as a path planner by making the robot to follow of path created by adding the force vector from obstacles and the goals. However, this method poses the problem of local minima where forces from obstacles balance each other, thus making the robot to stick in random locations.

Later on, more advanced version of potential field method that uses Harmonic Potential Function, which is based on solving a partial differential equation with Laplacian term is developed [21]. The biggest advantage of using harmonic potential functions is having only one local minimum at the goal point. Additionally, Goerzen *et*

al [21] stated that “it yields the maximum potential at the boundaries of obstacles, and has a non-degenerate Hessian at each critical point of the function.” Akishita et al. [22] and Connolly et al. [23] are the one published this method first, and Kim and Khosla [24] used the panel method for modeling obstacle in addition to harmonic potential function for path planning.

CHAPTER IV

RISK FACTOR ON MOTION PLANNING

Risk Analysis

Every activity involves some level of risk on certain action, and the meaning of risk is used differently by different disciplines [25]. Generally, risk is refers to an uncertainty and the result of uncertainty [26]. This usage of risk is used in the management side of a project to estimate the cost of any consequences, and corresponding safety procedure or contingency plan of a project is developed accordingly. For example, Perera [27] at NASA Johnson Space Center stated that NASA's risk management "includes processes to identify, analyze, plan, track, control, communicate and document risks." Risk Matrix is method that quantifies the likelihood and consequence of risk associated with action taken. Building a Risk Matrix begins with selecting levels of Probability of Failure and Consequence of action.

Table 1 is an example of level of probability, Table 2 is an example on consequence, or cost, of action, and Table 3 is the resulting Risk Matrix. The Probability and consequence of action is determined by the project designer after carefully reviewing similar cases from previous projects [27], and then Risk Matrix is tabulated by combining probability and cost of action together into one matrix form for future evaluation in the project. For example, the risk of an action that has probability of 3 and consequence of 5 has the highest risk according to the risk matrix, thus it should be avoided, and an alternate method must be implemented.

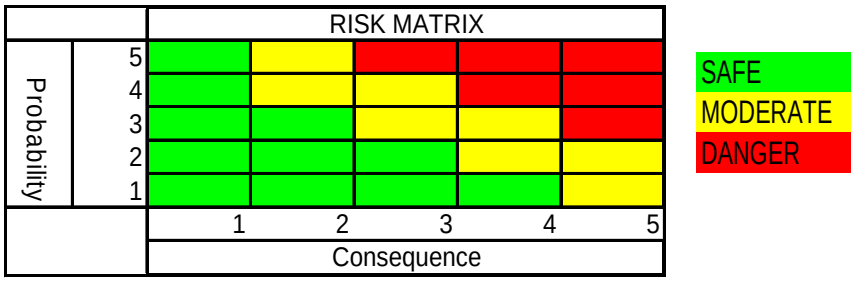
Table 1. Level of probability.

Probability	5	Very High
	4	High
	3	Moderate
	2	Low
	1	Very Low

Table 2. Cost of action on certain category.

		1	2	3	4	5
Cost	Category 1	No impact	Little Impact	Moderate Impact	Large Impact	Critical Impact
	Category 2	Negligible	Minor Damage	Damage	Major Damage	Critical Damage
	Category 3	No Injury	Few Injured	Injured	Many Injured	Tragedy
	Category 4	No Failure	Small Failure	Failure	Large Failure	Fiasco

Table 3. Risk matrix.



This Risk Analysis method must be converted into a form that can be used on motion planning problem. One way of applying risk analysis method on planning problem is defining risk mathematically, which is a product of probability and severity of incident [25],

$$\text{Risk} = \text{Probability} \times \text{Severity} \tag{5}$$

or, the Risk Factor [28] of an action is expressed as

$$RF = 1-(1-FI)(1-CI) \tag{6}$$

where RF is a Risk Factor of associate action, FI is a failure indices, and CI is a cost indices, respectively. Both FI and CI are value between 0 and 1, so that if both values are very low risk will go low, and vice versa.

Application of Risk Factor

The risk factor is used to create a property that resembles potential field method into the wavefront algorithm to create a path for a robot to follow, since direct application of one method to another is inappropriate because each methods represent working space differently. The wavefront algorithm is an algorithm with unit cost on each motion, and according to LaValle [11] “optimizes the number of stages to reach the goal.” The potential field method uses a potential function to represent features on the map, such as obstacle and goal, and create a guide for a robot. The risk factor is used as an additional cost of travelling on the grid of the wavefront algorithm that adjacent to an obstacle. This risk factor, as explained in previous section, is function of Euclidean distance between goal and the starting point.

In the RF calculation, failure index (FI) and cost index (CI) of each cell is proportional to the distance to the goal and position of each cell, and the distance to the starting point of robot and position of each cell, respectively. The reason is that the chance of failure when robot is located on the grid cell located near the starting point than the grid cell near the goal point. On the other hand, the cost of travelling is from the

starting point to the goal is opposite. That is, robot uses more resources to travel to the grid cell near the goal point that the grid cell nears the starting point, indicating that the cost increases as robot travels.

$$FI = \alpha D_{Goal} \quad (7)$$

$$CI = \beta D_{Start} \quad (8)$$

Creating path is performed by wavefront propagation algorithm on a map and selecting path is done by risk factor calculation on node. Below paragraphs are procedure of the algorithm.

- Calculate Risk Factor across map using equation (6)
- Perform wavefront expansion
- Add cost (using RF) on to wavefront expansion if the cell is neighboring obstacle. Otherwise, discard Risk Factor
- If the succession cell is neighboring to the cell with different potential, take the minimum value and propagate

Figure 11 is the pseudo code of an algorithm using risk factor. The overall process is similar to the wavefront algorithm, except at line 3, risk factor is used as an additional cost if the next available grid cell is an obstacle.

1. Assign 0 at the goal and ∞ on any obstacles
2. Any unexplored free grid cell has null value
3. At i^{th} iteration, assign $\phi(x) = i + \mathrm{RF}(x)$ into unexplored adjacent grid cell, where

$$RF(x) = \begin{cases} 1-\left(1-CI(x)\right)\left(1-FI(x)\right) & \text{Adjacent to an obstacle} \\ 0 & \text{Otherwise} \end{cases}$$

4. Assign the smallest cost by minimizing the potential, or $W_i = \min(\phi(x))$
5. If every grid cell is assigned then terminate; otherwise, let $i = i+1$ and move to the second step

Figure 11. Pseudo code of an algorithm using risk factor as an additional cost function.

Following example is a demonstration of the wavefront algorithm with risk factor calculation. For the simplicity of a calculation in the example, risk factor across the map is 1 everywhere.

1. Locate the goal and the initial position of the robot. This is the same as in the normal wavefront algorithm. See Figure 12.

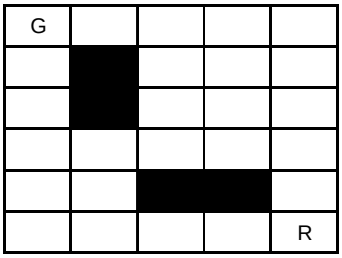


Figure 12. Initial configuration of map, which is the same as the wavefront algorithm.

2. Figure 13 shows the initial step of the algorithm. Since first available grid cells are adjacent to obstacle, their values are $W_1 = W_0 + 1 + RF = 0 + 1 + 1 = 2$. Proceed the process.

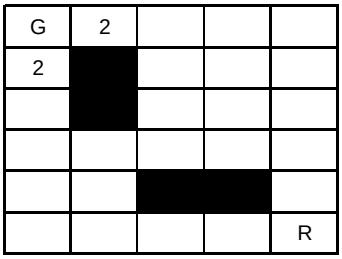


Figure 13. First propagation including risk factor.

3. Figure 14 demonstrates the how the algorithm proceed with different value of grid cell. If the unvisited grid cell is adjacent to the grid cell with two different values, take the minimum value and continue the process.

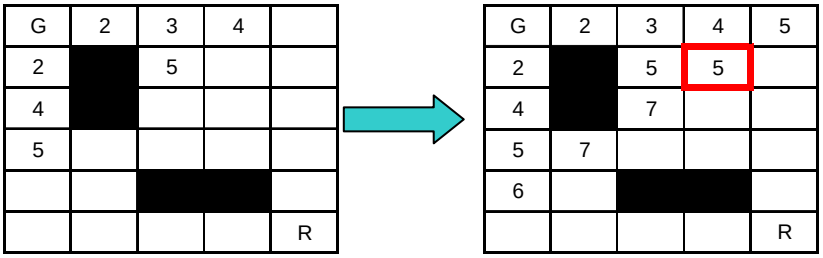


Figure 14. Wavefront expansion from the minimum value using risk factor calculation.

4. Figure 15 is the completed expansion of an algorithm using risk factor. Continue process until every unvisited grid cell is occupied. Terminate the process once every grid cell has assigned number. Grid cells that proceed with the minimum value of the predecessor grid cell are boxed with red rectangles.

G	2	3	4	5
2		5	5	6
4		7	6	7
5	7	9	8	8
6	8			10
7	8	10	11	R

Figure 15. Complete propagation with node expanded from the minimum value boxed in red.

CHAPTER V

SIMULATION AND TEST

The simulation of the method is performed prior to the actual implementation of robot to observe the behavior of the method. Both simulation and actual test is performed using MatLAB. The wavefront algorithm with risk factor calculation included will referred as risk factor method from now on.

Simulation Result

On Risk Factor calculation, FI and CI of each grid cells adjacent to the obstacle are calculated using (5) and (6), with proportionality constant 0.4 and 0.6, respectively. Risk Factor ranges from 0 to 1, with 1 being the highest value, and then divided into five levels for adding cost on wavefront propagation.

$$\begin{aligned} 0 < RF \leq 0.2 \quad & +1 \\ 0.2 < RF \leq 0.4 \quad & +2 \\ 0.4 < RF \leq 0.6 \quad & +3 \\ 0.6 < RF \leq 0.8 \quad & +4 \\ 0.8 < RF \leq 1 \quad & +5 \end{aligned}$$

Algorithms are tested on 5×5, 8×8, 12×12, and 6×8 size discrete map with 5%, 10%, 20% and 30% of node covered by obstacles, and compared to normal wavefront propagation.

Criteria of metrics on each method are measuring the number of turn, the distance traveled, and the number of obstacle robot passes during the navigation. The number of turn is measured by counting the number of direction change robot made

from the starting point to goal. The distance traveled is measured by the distance between each path; therefore, it is equal to the number of node robot passed. The number of obstacle is determined by the number of time robot passes a side of an obstacle. The number of turn is measured because less number of turn reduces position error of robot during the navigation, which can happen because of error accumulation from the encoder or any other sensor. Similarly, the distance travelled is measured to ensure the optimality of the path. The number of obstacle robot passes is measured because chance of collision increases with increase number of obstacle robot passes. See Appendix B for the simulation result.

Test Result

Figure 16 displays a physical set up for testing algorithm. Physical test of robot is performed in 6 × 6 tiled map as the maximum size because of space limitation. Each side of the map is 2m long and a lower left corner of the map is treated as an initial point. Robot's behavior is the same as the normal wavefront algorithm without any obstacle on the map. If robot detects any obstacle using its distance measuring sensor, position of obstacle is updated on robot's map if robot does not have any information about the obstacle prior to navigation. Once update is complete, robot re-creates a path to the goal with the current position as the beginning point. In this test, the initial point and goal point are assumed as known

Overall behavior of robot is the same as in the simulation. The difference was priori information. In the simulation, every position of obstacle is known; however, in the real test, robot must update its map as it operates. Although the final path robot take

was not exactly the same as in the simulation, overall behavior of moving around an obstacle, avoiding obstacle crowded area was observed.

Figure 16. Testing performance of algorithm using mobile robot platform in the 6 x 6 tile map with the total length of 2m. The distance between each tick mark is 1m.

CHAPTER VI

DISCUSSION

Upon the completion of the wavefront propagation on the map, simulation is programmed to take the grid cell with the shortest Euclidean distance if the assigned value is the same among the available grid cell for both methods for the consistency on the performance evaluation; otherwise, grid cell with the lowest among all is selected. The result of the simulation on both wavefront algorithm with risk factor method and normal wavefront algorithm displays same number of grid cells robot travels from the starting point to the goal point. However, there are distinct differences on the number of turn and the number of obstacle robot face, and the shape of the resulting path.

While both method moves same number of grid cell from the starting to the goal point, the wavefront algorithm with risk factor method turns and faces obstacles less than the path created using the normal wavefront algorithm. Additionally, path from the risk factor method moves around the obstacle with maintaining certain distance, yet keeping the total travelling distances minimum. These trends are clear as the size of the map and the percentage of obstacle present on the map increases.

For example, on 8×8 map with 30% of map occupied by obstacle on Figure B-2 (c), path from the risk factor method turns once with passing one side of an obstacle. On the contrary, normal wavefront method turns 9 times while passing a side of an obstacle 8 times. Most importantly, their paths are completely different. For example, the path from the risk factor method on Figure B-1(c), Figure B-2 (b), (c) and (d), Figure B-3(b)

and (c), Figure B-4 avoid obstacle crowed area as much as possible, where as the normal wavefront method moves anyway. Similar behavior appears on the different results and if the presence of obstacle on map is zero, then the path for both methods is the same. From this result, the greatest benefit of using risk factor is creation of safe path without losing the optimality.

Optimality in Dynamic Programming Perspective

The proof of optimality of Dijkstra's Algorithm in the Dynamic Programming perspective and how the wavefront algorithm complies with the Principle of Optimality is demonstrated in Chapter 2. As an algorithm based on the wavefront algorithm, the newly created method also must comply with the Principle of Optimality.

The Principle of Optimality optimizes process by maximizing reward or minimizing cost of process. For the case of shortest path finding, the optimal path is the one that has the lowest cost among all possible routes. Usage of the risk factor in the wavefront algorithm still preserves this property. Recall that the cost function in the algorithm produces those found in the wavefront algorithm and the risk factor as an additional cost if the grid is adjacent to an obstacle. Once cost is measured, assign the minimum value to the next available grid cell. This added value increases cost around the obstacle and affect its neighboring grid cell in the further execution. The optimal path is one with minimal inclusion of risk factor throughout execution.

Distinction from the Gradient Method

The risk factor method uses wavefront algorithm as a base and risk factor to calculate the cost of travel, which is calculated using two indices. Each index is function

of distance. Cost index is proportional to the distance between starting point of robot and individual grid cell, because cost increases as robot move away from its initial position. Failure index is proportional to the distance between goal and individual grid cell because chances of failure decreases as robot move towards the goal.

Unlike the gradient method, where intrinsic cost is measured in everywhere on the map, only the risk factor of the grid cell adjacent to obstacle is counted towards the cost function because other node is free and does not pose any severe danger.

CHAPTER VII

CONCLUSION AND FUTURE WORK

Conclusion

In this thesis application of risk factor on path planning of mobile robot is introduced and its performance is evaluated. Referring to simulation results in appendix, the risk factor method enhances safety of robot on path planning by moving around an obstacle with maintaining certain distance, and avoids moving through the area with large obstacle occupation. Path from normal wavefront propagation move through the area even if the presence of obstacle is large, in which increases chances of collision with obstacle on the path.

Risk factor method has a tendency of moving through obstacle-free area, yield better result on safety on simulation result while keeping the distance travel the minimum and the performance is confirmed on the real robot test. Therefore, the conclusion for this work is that risk factor method is more preferable than normal wavefront method for the success of navigation.

Future Work

The simple method of calculation each indices of risk factor and the map representation is used for calculation simplicity. Following list is the possible future work for an improvement.

- Usage of fuzzy logic or statistical approach on the risk factor calculation
- Increased resolution on the map for better representation

- More precise distance measuring sensor on mobile robot

REFERENCES

[1] J.A. Cobano, R. Ponticelli and P. Gonzalez de Santos, "Mobile robotic system for detection and location of antipersonnel land mines: field tests," *Industrial Robot: An International Journal*, Vol. 35 Iss: 6, 2008, pp.520 – 527.

[2] I.J. Cox and G.T. Wilfong, *Autonomous Robot Vehicles*, New York; Springer–Verlag, 1990.

[3] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *IEEE International Conference on Robotics and Automations*, St. Louis, MO, March 25 – 28, 1985, pp. 500 – 505.

[4] S. Nazari, M.R. Meybodi, M.A. Salehigh, and S. Taghipour. "An Advanced Algorithm for Finding Shortest Path in Car Navigation System," *First International Conference on Intelligent Networks and Intelligent Systems,* 2008, pp.671-674.

[5] S. Robinson. (2004). Mapping Magic [Online]. Available: http://www.siam.org/news/news.php?id=255

[6] Acroname Robotics. (2011). Sharp IR Rangers Information [Online]. Available: http://www.acroname.com/robotics/info/articles/sharp/sharp.html

[7] *Sharp GP2D12 Optoelectronic Device datasheet*, Sharp Corp., Camas, WA, 2005, pp. 4.

[8] J. M. Esposito and O. Barton. (2011). Matlab Toolbox for the iRobot Create [Online]. Available: http://usna.edu/Users/weapsys/esposito/roomba.matlab/

[9] J. Dixon and O. Henlich. (1997). Mobile Robot Navigation [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd/

[10] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton, NJ; Princeton University Press, 1962.

[11] S. M. LaValle, *Planning Algorithms.* Cambridge, NY: Cambridge University Press, 2006.

[12] R. E. Larson and J. J. Casti, *Principles of Dynamic Programming*: *Part I Basic Analytic and Computational Methods*. New York: Marcel Dekker, Inc., 1978.

[13] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ: John Wiley & Sons, 2007.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[15] E. W. Dijkstra. (1959, December 01). A note on two problems in connexion with graphs. *Numerische Mathematik.* [Online]. *1(1)*, pp. 269–271. Available: http://dx.doi.org/10.1007/BF01386390

[16] Algolist.com. (2010). Dijkstra's Algorithm [Online]. Available: http://www.algolist.com/Dijkstra%27s_algorithm

[17] P. Varshavskaya. (2009). Comp150-07: Intelligent Robotics [Online]. Available: http://www.cs.tufts.edu/comp/150IR/labs/wavefront.html

[18] R. B. Muhammad. (2010). Breadth First Search Traversal Algorithm [Online]. Available: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/breadthSearch.htm

[19] Society of Robots. (2007). WAVEFRONT ALGORITHM [Online]. Available: http://www.societyofrobots.com/programming_wavefront.shtml

[20] K. Konolige, "A Gradient Method for Realtime Robot Control," *Proc. IEEE/RSJ Conf. Intell. Robots Syst.*, Takamatsu, Japan, 2000, pp. 639–646.

[21] C. Goerzen, Z. Kong, and B. Mettler, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *J. Intell. Robot. Syst.*, Vol 57. pp. 65-100. 2010.

[22] S. Akishita, S. Kawamura, and K. Hayashi, "Laplace potential for moving obstacle avoidance and approach of a mobile robot," *Proceedings of the Japan-USA Symposium on Flexible Automation* (a Pacific Rim Conference), 1990, pp. 139-142.

[23] C. I. Conolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," *Proc. IEEE Int. Conf. on Robotics and Automation*, 1990, pp. 2102-2106.

[24] J. Kim and P. Khosla, "Real-Time Obstacle Avoidance Using Harmonic Potential Function," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, California, 1991, pp. 790-796.

[25] E. A. C. Crouch and R. Wilson, *Risk/Benefit Analysis*. Cambridge, MA: Ballinger Publishing Company, 1982.

[26] D. B. Hertz, *Risk Analysis and its Applications*. New York: John Wiley & Sons, 1983.

[27] J. Perera and J. Holsomback, "An integrated risk management tool and process," *IEEE Aerospace Conference*, 2005, pp.129-136.

[28] H. Eisner, *Computer Aided Systems Engineering*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[29] L. Goodman, A. Lauschke and E. W. Weisstein. (2011). Dijkstra's Algorithm [Online]. Available: http://mathworld.wolfram.com/DijkstrasAlgorithm.html

APPENDIX A

EXAMPLE OF DIJKSTRA'S ALGORITHM

Dijkstra's Algorithm is an algorithm for finding the single source shortest path in graph [11], [16]. It functions by constructing a shortest-path tree from the initial vertex to every other vertex in the graph [29], where every edge has non-negative cost. This example is mean to help understand procedure and the application of Dijkstra's algorithm in Chapter II can be used to find the shortest path in the network.

Consider node and edge network given in Figure A-1 below. Starting at the red node, we would like to find the shortest path to reach the green node. The cost of travelling from each node is given.

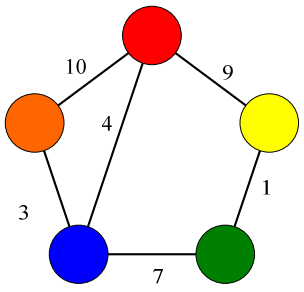


Figure A-1. Node-edge network for the shortest path problem.

The first step is assigning zero on the initial node and infinity on the rest of the node, as in Figure A-2. Then search for unvisited node adjacent to the initial node, which are orange, blue, and yellow node. Add the value of the initial node and the cost of edge,

select one with the lowest value, and assign resulting value to the target node. Mark red node as visited (change color to grey in this example).

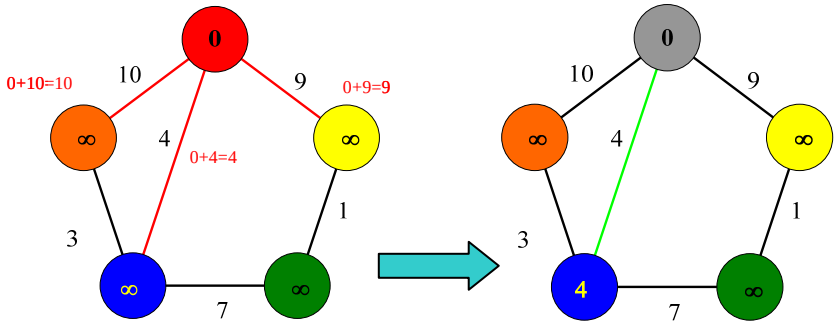


Figure A-2. First iteration to find the node with the minimum cost, where minimum cost is obtained by summing the value of node and the cost of edge connecting node. Select the edge that connects the node with the lowest total value.

Set blue node as the current node and repeat the process again to search for the unvisited node, as in Figure A-3. Path from blue node to orange node is the minimum among all choice. Select edge connecting blue to orange node and set orange node as visited.

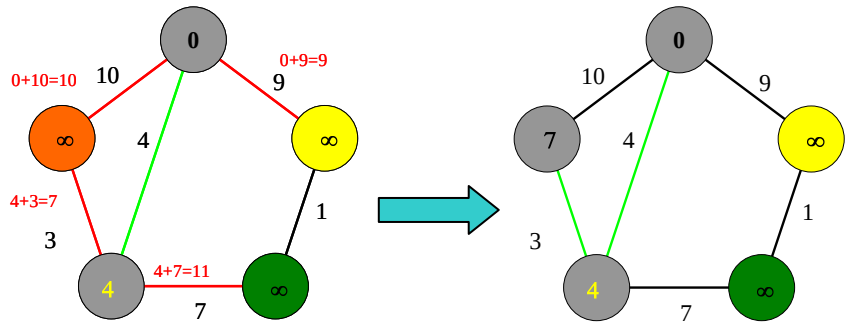


Figure A-3. Repetition of the process. Search for unvisited node, compare the cost, and select the node with the lowest total cost.

Repeat the process to find the minimum cost. The initial (red) node to yellow node has the minimum cost, so select edge connecting the initial node and yellow node. Mark yellow node as visited. See Figure A-4.

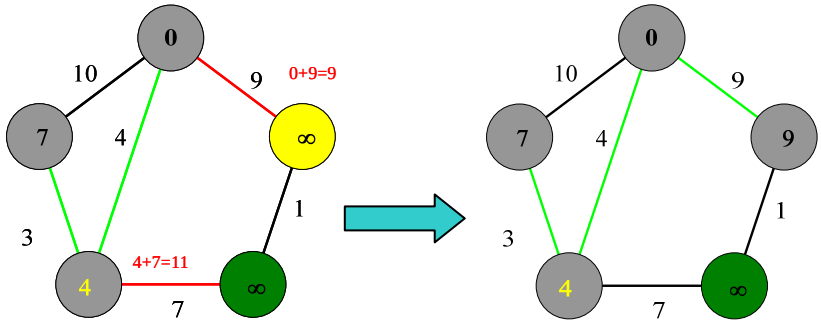


Figure A-4. Repetition of the process.

Repeat the process for the green node. Edge connecting yellow and green node is lower than the edge connecting blue and green node; thus, select the edge between yellow and green node and mark green node edge as visited. Terminate the process since every node is visited. See Figure A-5.

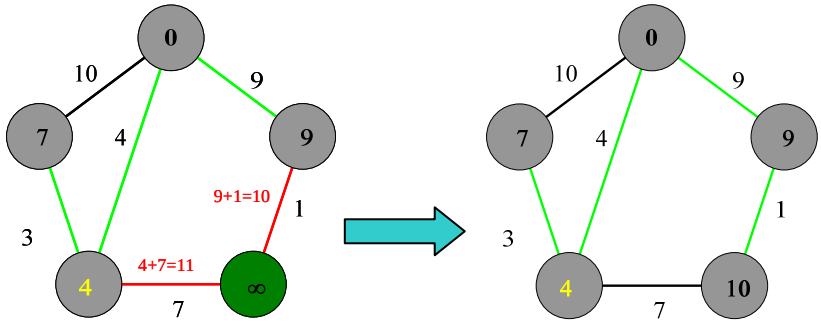


Figure A-5. The last process of iteration. Terminate the process if every node is visited.

Figure A-6 is the network of the shortest path using Dijkstra's algorithm. The complete network of the shortest path to the every node from red node is represented with green edge. The shortest path from red node to green node is via yellow node with the total cost of 10.

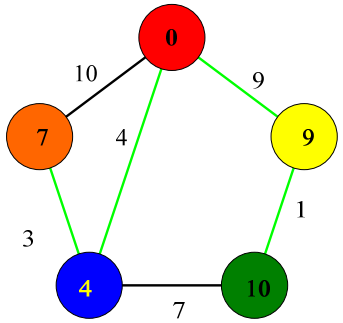


Figure A-6. The shortest path network created by Dijkstra's Algorithm.

APPENDIX B

SIMULATION RESULT

This appendix contains the simulation result of the wavefront algorithm with risk factor method. Left side is the wavefront algorithm with risk factor included and the right side is the normal wavefront algorithm. In this simulation, value of goal and free empty grid cell are 1 and 0, respectively.

5 × 5 Map

RF Included

5	4	3	2	**G**
6	5	6	3	2
7	8		6	3
8	9	8	5	4
S	8	7	6	5

Turn 3
Dist 7
Obs 0

Normal

5	4	3	2	**G**
6	5	4	3	2
7	6		4	3
8	7	6	5	4
S	8	7	6	5

Turn 5
Dist 8
Obs 2

(a) 5% of map covered by Obstacle

15	16	13		**G**
14		10	5	2
11	14		6	3
10	9	8	5	4
S	8	7	6	5

Turn 3
Dist 8
Obs 1

7	6	5		**G**
8		4	3	2
9	8		4	3
8	7	6	5	4
S	8	7	6	5

Turn 4
Dist 8
Obs 4

(b) 10% of map covered by Obstacle

Figure B-1. Simulation result for 5x5 map size (a) 5% of grid covered with obstacle. (b) 10% of grid covered with obstacle. (c) 20% of grid covered with obstacle. (d) 30% of grid covered with obstacle.

7	6	5	2	G
8	9		5	4
9	10	15	10	
10	13			
S	14	17	20	23

Turn 1
Dist 8
Obs 1

5	4	3	2	G
6	5		3	2
7	6	5	4	
8	7			
S	8	9	10	11

Turn 5
Dist 8
Obs 6

(c) 20% of map covered by Obstacle

	8	5	2	G
14	11		5	4
17	14		10	
	19		17	
S	20	25		0

Turn 2
Dist 8
Obs 6

	4	3	2	G
6	5		3	2
7	6		4	
	7		5	
S	8	9		0

Turn 2
Dist 8
Obs 6

(d) 30% of map covered by Obstacle

Figure B-1 Continued.

8 × 8 Map

RF Included

12	11	13		7	3	2	**G**
11	10	9	9	5	4	3	2
10	9	8	7	6	8	4	3
11	10	9	11	10		8	4
12	11	13		13	10	6	5
13	12	13	13	9	8	7	6
14	13	12	11	10	9	8	7
S	14	13	12	11	10	9	8

Turn 9
Dist 14
Obs 0

Normal

10	9	8		4	3	2	**G**
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
11	10	9	8	7		5	4
12	11	10		8	7	6	5
13	12	11	10	9	8	7	6
14	13	12	11	10	9	8	7
S	14	13	12	11	10	9	8

Turn 11
Dist 14
Obs 4

(a) 5% of map covered by Obstacle

8	7	6	5	4	3	2	**G**
9	8	10	6	5	4	3	2
10	12		10	9	5	4	6
11	15		17		9	11	
12	13	17	17	17	13		26
13	14	15	16	15	14	18	19
14	15	16	17	16	18	19	20
S	16	17	18	20		23	21

Turn 3
Dist 14
Obs 0

8	7	6	5	4	3	2	**G**
9	8	7	6	5	4	3	2
10	9		7	6	5	4	3
11	10		8		6	5	
12	11	10	9	8	7		11
13	12	11	10	9	8	9	10
14	13	12	11	10	9	10	11
S	14	13	12	11		11	12

Turn 11
Dist 14
Obs 5

(b) 10% of map covered by Obstacle

Figure B-2. Simulation result for 8x8 map size (a) 5% of grid covered with obstacle. (b) 10% of grid covered with obstacle. (c) 20% of grid covered with obstacle. (d) 30% of grid covered with obstacle.

8	7	6	5	4	3	2	**G**
12	8	7	6	8	4	3	2
	12	8	10		11	4	3
	16	9	13	20		8	4
30		19		14	10	6	5
23	28		22	12	11	7	6
22	21	26		19		11	7
S	20	19	18	14	13	9	8

Turn 1
Dist 14
Obs 2

8	7	6	5	4	3	2	**G**
9	8	7	6	5	4	3	2
	9	8	7		5	4	3
	10	9	8	9		5	4
18		10		8	7	6	5
17	16		10	9	8	7	6
16	15	14		10		8	7
S	14	13	12	11	10	9	8

Turn 4
Dist 14
Obs 6

(c) 20% of map covered by Obstacle

11	10	9	5	4	3	2	**G**
12	14		12	8	4	3	2
13	17				8	7	3
14	15	22	23	16	12		7
15	19			20	16	15	11
16	17	21	28	24		25	
17	18	22		28	38		
S	19	20	24	28		0	0

Turn 1
Dist 14
Obs 1

8	7	6	5	4	3	2	**G**
9	8		6	5	4	3	2
10	9				5	4	3
11	10	9	8	7	6		4
12	11			8	7	6	5
13	12	11	10	9		7	
14	13	12		10	11		
S	14	13	12	11		0	0

Turn 9
Dist 14
Obs 8

(d) 30% of map covered by Obstacle

Figure B-2 Continued.

12 × 12 Map

RF Included

16	15	14	13	8	7	6	5	4	3	2	**G**
17	16	19		13	8	7	6	5	4	3	2
16	15	14	15	10	9	8	7	6	5	4	3
15	14	13	12	11	10	9	8	7	6	5	4
16	15	14	13	12	11	10	9	8	7	6	5
17	16	15	18	13	12	15	10	9	8	7	6
18	17	20		18	17		19	14	9	8	7
19	18	19	23	19	18	25			14	9	8
20	19	20	19	18	17	16	19	16	11	10	9
21	20	22	18	17	16	15	14	13	12	11	10
22	25		23	22	17	16	15	14	13	12	11
S	26	31	28		22	17	16	15	14	13	12

Turn 13
Dist 22
Obs 0

Normal

12	11	10	9	8	7	6	5	4	3	2	**G**
13	12	11		9	8	7	6	5	4	3	2
14	13	12	11	10	9	8	7	6	5	4	3
15	14	13	12	11	10	9	8	7	6	5	4
16	15	14	13	12	11	10	9	8	7	6	5
17	16	15	14	13	12	11	10	9	8	7	6
18	17	16		14	13		11	10	9	8	7
19	18	17	16	15	14	15			10	9	8
20	19	18	17	16	15	14	13	12	11	10	9
21	20	19	18	17	16	15	14	13	12	11	10
22	21		19	18	17	16	15	14	13	12	11
S	22	21	20		18	17	16	15	14	13	12

Turn 21
Dist 22
Obs 6

(a) 5% of map covered by Obstacle

17	16	15	14	13	12	11	10	9	8	7	**G**
18	17	16	15	14	13	12	11	10	14		7
19	18	17	20	15	14	13	12	11	16		13
20	19	22		24	19	14	13	12	13	22	14
21	20	21	30			19	14	13	18		20
22	21	22	23	26	21	16	15	14	15	20	19
23	22	27	24	23	18	17	16	15	16	17	18
24	27		31		27	18	17	20	17	18	19
25	26	30	30	35		23	22		22	19	20
26	27	31	29	28	29	24	23	28	23	24	25
27	32		32	27	26	25	24	25	28		
R	33	34	29	28	27	26	25	26	27	32	38

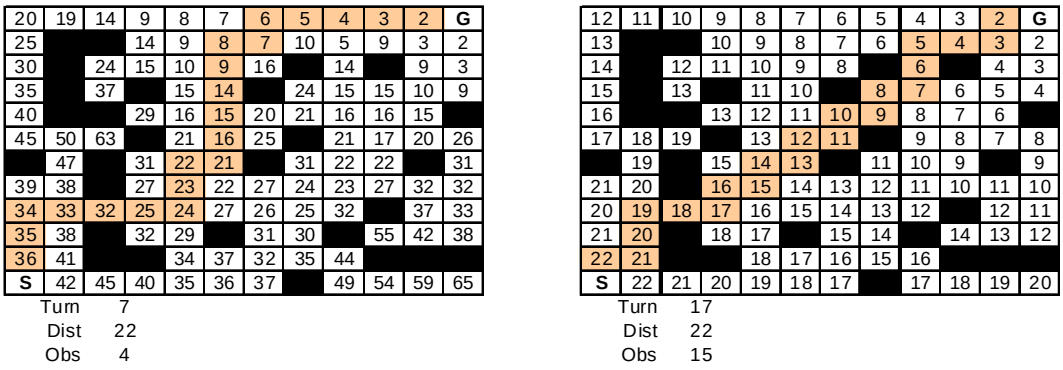
Turn 7
Dist 22
Obs 1

12	11	10	9	8	7	6	5	4	3	2	**G**
13	12	11	10	9	8	7	6	5	4		2
14	13	12	11	10	9	8	7	6	5		3
15	14	13		11	10	9	8	7	6	5	4
16	15	14	15			10	9	8	7		5
17	16	15	14	13	12	11	10	9	8	7	6
18	17	16	15	14	13	12	11	10	9	8	7
19	18		16		14	13	12	11	10	9	8
20	19	18	17	18		14	13		11	10	9
21	20	19	18	17	16	15	14	13	12	11	10
22	21		19	18	17	16	15	14	13		
R	22	21	20	19	18	17	16	15	14	15	16

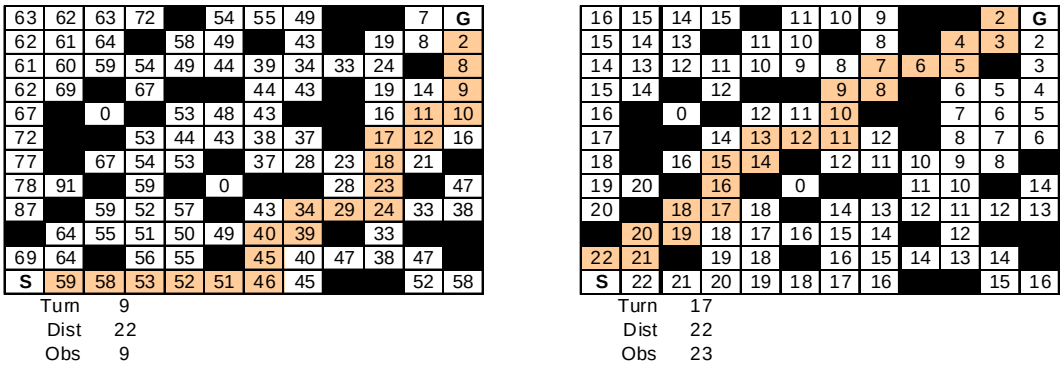
Turn 17
Dist 22
Obs 11

(b) 10% of map covered by Obstacle

Figure B-3. Simulation result for 10x10 map size (a) 5% of grid covered with obstacle. (b) 10% of grid covered with obstacle. (c) 20% of grid covered with obstacle. (d) 30% of grid covered with obstacle.



(c) 20% of map covered by Obstacle



(d) 30% of map covered by Obstacle

Figure B-3 Continued.

6 × 8 Map

RF Included

11	10	9	8	7	6	2	**G**
12	11	10	9	11		6	2
13	12	11	10	11	11	4	3
14	13	12	11	13		8	4
14	13	12	11	10	10	6	5
S	12	11	10	9	8	7	6

Turn 3
Dist 12
Obs 0

Normal

8	7	6	5	4	3	2	**G**
9	8	7	6	5		3	2
10	9	8	7	6	5	4	3
11	10	9	8	7		5	4
12	11	10	9	8	7	6	5
S	12	11	10	9	8	7	6

Turn 8
Dist 12
Obs 4

(a) 5% of map covered by Obstacle

8	7	6	5	4	3	2	**G**
9	8	7	9	5	4	3	2
10	9	12		9	5	4	6
11	12		12	7	6	11	
12	13	14	11	10	10		26
S	14	15	14		14	18	19

Turn 5
Dist 12
Obs 0

8	7	6	5	4	3	2	**G**
9	8	7	6	5	4	3	2
10	9	8		6	5	4	3
11	10		8	7	6	5	
12	11	10	9	8	7		11
S	12	11	10		8	9	10

Turn 10
Dist 12
Obs 4

(b) 10% of map covered by Obstacle

14	13	12	8	7	3	2	**G**
15	17		15		7	6	2
16	17	20	18		14		6
17	18	19	21	24	21		10
18	19	22		29			17
S	20	21	24	25	29	36	

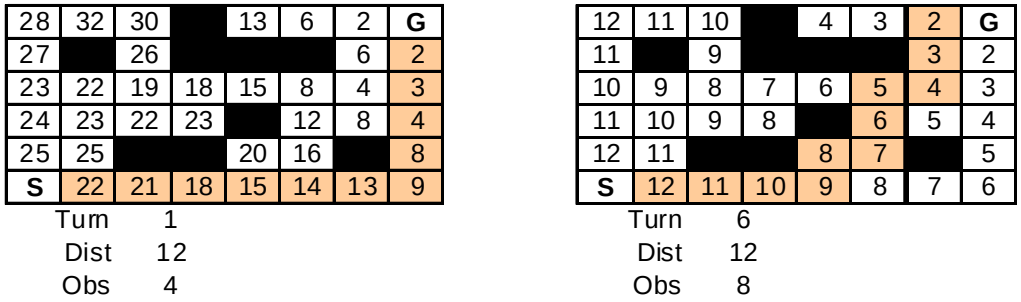
Turn 1
Dist 12
Obs 2

8	7	6	5	4	3	2	**G**
9	8		6		4	3	2
10	9	8	7		5		3
11	10	9	8	7	6		4
12	11	10		8			5
S	12	11	10	9	10	11	

Turn 6
Dist 12
Obs 9

(c) 20% of map covered by Obstacle

Figure B-4. Simulation result for 6x8 map size (a) 5% of grid covered with obstacle. (b) 10% of grid covered with obstacle. (c) 20% of grid covered with obstacle. (d) 30% of grid covered with obstacle.



(d) 30% of map covered by Obstacle

Figure B-4 Continued.

VITA

Sung Huh received his Bachelor of Science degree in mechanical engineering from Texas A&M University in 2008 and he continued to study for his Master of Science degree in mechanical engineering, with concentration in dynamics and control field. He received his Master of Science degree in 2011. His research interests are dynamics and control, manipulator robotics, and mobile robot. He can be reached at 200 MEEN Building, 3123 TAMU, College Station, TX 77840.